Domain Administration of Task-role Based Access Control for Process Collaboration Environments

Yahui Lu  
College of Software  
Shenzhen University  
Shenzhen 518060, P.R.China  
luyahui@szu.edu.cn

Li Zhang  
School of Software  
Tsinghua University  
Beijing 100084, P.R.China  
zhangli@tsinghua.edu.cn

Abstract—The fast evolving workflow technologies facilitate organizations to interact and cooperate with each other to achieve their business goals by process collaborations. Task-role based access control is an important security mechanism to protect data and resources in information systems. However, the traditional centralized authorization and administration mechanism in access control can not satisfy the administrative requirements in process collaboration environments. In this paper, we propose a domain based administration model for task-role based access control (DATRBC), in which the authorization and administration permissions are distributed to multiple administrative domains and administrative roles. Then we propose the solution to detect and resolve the conflicts between access control policies defined by different administrative roles. We also described the implementation of the model in the PLM product and the experiments based on the practical application data.

Keywords—Access Control; Workflow; Process Collaboration; Administrative Role; Administrative Domain

I. INTRODUCTION

Workflow becomes a promising solution for organizations to automate their business processes. There is a trend to utilize workflow technologies to integrate applications within and across organizational boundaries [1]. In a process collaboration environment, several business processes which may locate in different organizations can interact with each other to achieve the common business goals of these organizations. The resources in each participants are needed to protected in such process collaboration environment. Access control is a fundamental security mechanism to protect data and resources in information systems. The most widely used access control models in workflow systems are RBAC [4], TBAC [5] and TRBAC [6]. Task-role based access control model (TRBAC) [6] combines roles and tasks together to support both static and dynamic access control in workflow and non-workflow enterprise environments. It can be used as the basic access control mechanism in process collaboration systems. Many access control models have been proposed for collaborative systems [2], [3], [11].

In another view, the elements of these access control models, such as roles, users, tasks, permissions and their relationships, also need to be managed by access control mechanism in order to guarantee the system security. In traditional access control models, all the authorization and administration permissions are assigned to one global system administrator and managed by one small administrative team. This centralized administration can not flexibly manage the specific access control requirements of each department or project team. Several administrative models have been proposed to decentralize the administrative privileges of administrators [7], [8], [9], [10], [11]. However, these research mainly focus on the management of RBAC model within one organization. For process collaboration environments, it is still a challenging goal to decentralize the details of administration on TRBAC model without loosing central control over broad policies.

In this paper, we propose the DATRBC model, a domain based administration model for task-role based access control for process collaboration environments. In DATRBC, the authorization and administration permissions are distributed to multiple administrative domains. An administrative domain can be divided into several sub-domains. Several administrative domains can be combined together to form a larger united administrative domain. Each administrative domain is assigned with an administrative role. This administrative role can only execute the specified authorization and administration operations within its administrative domain. We propose an algorithm to detect the conflicts during the stage of policy definition and propose several different strategies to resolve these conflicts during the stage of run time access check. By introducing the concept of administrative domain and administrative role, the DATRBC model can flexibly meet the administrative requirements in process collaboration environments.

II. DATRBC MODEL

A. Definition of DATRBC model

Definition 1: As figure 1 shows, the DATRBC model has the following components:
(1) $U$, $R$, $T$, $OBJ$, and $OP$: the set of users, roles, tasks, objects, and object operations, respectively.
The DATRABAC are composed by four parts: authorization, administration, access check, and constraints. In authorization part, object permissions are defined as the operations on the objects. These permissions can be assigned to tasks, roles, and users to support the different access control requirements. User can get these permissions by UPA directly, or by its roles (through URA, RH and RPA), or by the tasks that it possesses dynamically during the process executions (through URA, RH, RTA and TPA). In administration part, the authorization and administration permissions are defined as the authorization and administration operations on administrative objects. The administrative domains are composed by these authorization and administrative permissions. Each administrative domain is assigned with an administrative role by ARDA. A user can manage the administrative domain if it is assigned with the administrative role by UARA. In access check part, the function Ispermit is to decide whether user’s permission is permitted or denied.

B. Administrative domain and administrative role

The administrative domain includes administrative objects and the authorization and administration operations on these administrative objects. The administrative object includes users, roles, tasks, permissions, and child administrative roles.

Example 1: The administrative role on development department XmDAR can be defined as the following.

\[ \text{XmDAR.AD.AOP} = \{ \text{ModifyAdminDomain, AssignUserRole, GrantRolePerm, GrantUserRole} \} \]

AND \( \text{XmDAR.AD.AO.AR} = \{ \text{XmDG1AR} \} \)

AND \( \text{XmDAR.AD.AO.U} = \{ \text{factory= "XiaMen" and department="design"} \} \)

AND \( \text{XmDAR.AD.AO.R} = \{ \text{Designer, Auditor} \} \)

AND \( \text{XmDAR.AD.AO.P} = \{ \text{DesignFile, Browse}, \text{DesignFile, CheckIn}, \text{ProductManual, Browse} \} \)

Since junior administrative roles can be included in the administrative object of senior administrative role, a partially ordered administrative role hierarchy can be defined by the administrative domains.

Definition 2: Administrative Role Hierarchy(ARH): For administrative domain XmDAR can be defined as the following. XmDAR.AD.AOP = \{ ModifyAdminDomain, AssignUserRole, GrantRolePerm, GrantUserRole \}

AND \( \text{XmDAR.AD.AO.AR} = \{ \text{XmDG1AR} \} \)

AND \( \text{XmDAR.AD.AO.U} = \{ \text{factory= "XiaMen" and department="design"} \} \)

AND \( \text{XmDAR.AD.AO.R} = \{ \text{Designer, Auditor} \} \)

AND \( \text{XmDAR.AD.AO.P} = \{ \text{DesignFile, Browse}, \text{DesignFile, CheckIn}, \text{ProductManual, Browse} \} \)

Since junior administrative roles can be included in the administrative object of senior administrative role, a partially ordered administrative role hierarchy can be defined by the administrative domains.

Definition 2: Administrative Role Hierarchy(ARH): For administrative role \( ar_i \) and \( ar_j \), if \( ar_j \) belongs to the administrative domain of \( ar_i \), we call \( ar_i \) is the parent administrative role of \( ar_j \). Formally,

\[ \text{ARH} = \{ (ar_i, ar_j) | \exists ad \in AD((ar_i, ad) \in ARDA) \land (ar_j \in ad) \} \]

The administrative domains can also be united to form a new administrative domain.

Definition 3: Union of administrative domain: For administrative domain \( ad_1, ad_2, \ldots, ad_n \), the union of them is defined as

\[ U_{ad} = \bigcup_{1 \leq i \leq n} ad_i = \bigcup_{1 \leq i \leq n} ad_i.AO \times ad_i.AOP \]

The permissions of each administrative role can be easily changed by modifying its administrative domain. By the hierarchy of administrative domains and administrative roles,
the administrative domain can be divided into several subdomains. Several administrative domains can be combined together to form a larger united administrative domain. Further more, because each administrative role can only execute the specified operations within its domain, even if this administrative role executes wrong operations, it can not influence users or permissions beyond its domain. Thus, the DATRBAC model provides a safer mechanism for authorization and administration in access control.

C. Conflicts of access control policies

The access control policy is defined by administrative role. Since there may exist several administrative roles in DATRBAC model, different access control policies may be defined by these administrative roles for the system.

Definition 4: An access control policy defined by an administrative role is a tuple (URA, RH, RTA, RPA, TPA, UARA, ARDA, C). The access control policy can be divided into two parts: the authorization policy and the administration policy.

Definition 5: An authorization policy defined by an administrative role is a tuple (UPA, URA, RH, RTA, RPA, TPA, C). Authorization policy defines the authorization rules to decide the user’s permissions to objects in run-time access check.

Definition 6: An administration policy defined by an administrative role is a tuple (UARA, ARDA, C). Administration policy defines the administration rules to decide the administrative role’s permissions to the administrative objects.

Example 2: An authorization policy P1 by XmDAR can be defined as the following.
P1 := (UPA{(Anyone, Browse, ProductManual, Permit),
(Wang, CheckIn, DesignFile, Deny)},
URA(Wang, Designer),
RPA{(Designer, Browse, DesignFile, Permit),
(Guest, Browse, DesignFile, deny)})

Example 3: An authorization policy P2 by XmDPAR can be defined as the following.
P2 := (RTA(Designer, Design),
TPA{(Design, CheckIn, DesignFile, Permit),
(ManufacturePartS, Browse, DesignFile, Permit)})

From the examples above, we find that the administrative domains of administrative roles may overlap with each other. Different administrative roles may have the same user (or role, task, administrative role, permission etc) as their administrative objects. Because each administrative role can define its own access control policy and the system access control policy includes all these policies, there may exist conflicts of these access control policies. For example, user Wang does not have the permission to check in the design file by UPA of policy P1. And Wang may have the permission to check in the design file in the execution of Design task by rule (Wang, Designer) in URA of policy P1, rule (Designer, Design) in RTA and (Design, CheckIn, DesignFile, Permit) in TPA of policy P2. There may exist a conflict to decide whether Wang has the permission to check in the design file in the run time access check.

These conflicts can be detected and resolved during two stages: the definition of policies (design time) or access check (run time).

1) In design time, after administrative roles finish the definition of access control policies, a conflict detection algorithm can be called to detect the potential conflicts of these policies. If the potential conflicts are detected out, the system can notify the related administrative roles to change their policies. These conflicts can also be resolved by system in run time access check if these administrative roles insist on their policies and do not change the conflict rules. Algorithm 1 is to detect the potential conflicts of access control policies. It first merges all the rules into one access control policy. From the examples above, we find that the administrative roles may have the same operations, e.g. design file by UPA of policy P1. And Wang may have the permission to check in the design file by UPA of policy P2. There may exist a conflict to decide whether Wang has the permission to check in the design file in the run time access check.

Algorithm 1: Conflict Detection Algorithm
Input: access control policy P1 by ar1, P2 by ar2, ..., Pn by arn
Output: the conflict rules set Sc
1. P = ∪i≤n Pi = {∪i≤n P.UPA, ∪i≤n P.URA,
∪i≤n P.RH, ∪i≤n P.RTA, ∪i≤n P.RPA, ∪i≤n P.TPA,
∪i≤n P.UARA, ∪i≤n P.ARDA, ∪i≤n P.C} 
1) //Detect conflicts for administrative roles
2. For each (ar, ad) ∈ P.ARDA
3. if exist (ar, ad′) ∈ P.ARDA ∧ ad′ ≠ ad′
4. S = S ∪ {(ar, ad), (ar, ad′)}
1) //Detect conflicts for administrative roles
5. Compute InDegree of each role vertex in P.RH
6. While(exist role r with InDegree(r)=0)
7. For each (r, r′) ∈ P.RH
8. P.RH = P.RH ∪ {r, r′}
9. InDegree(r′) = InDegree(r′) − 1
10. S = S ∪ P.RH
1) //Detect permission conflicts of tasks
11. For each (x, p, y) ∈ P.TPA
12. For each (x, p, y) ∈ P.TPA
13. if x ≠ y then
14. S = S ∪ {(x, p, y)}
13. //Detect permission conflicts of roles
15. For each (r, x, p) ∈ P.RPA
16. For each (r′, y, p) ∈ P.RPA
17. if x ≠ y ∧ r′ ≤ r then
18. S = S ∪ {(r, p, x), (r′, p, y)}
19. For each (x, p, y) ∈ P.TPA
20. For each (r, t) ∈ P.RTA
21. if x ≠ y ∧ r′ ≤ r then
22. S = S ∪ {(x, p, x), (r′, t), (t, p, y)}
23. For each (u, p, x) ∈ P.UPA
For each \((u, p, y) \in P.UPA\)

If \(x \neq y\) then \(S = S \cup \{(u, p, x), (u, p, y)\}\)

For each \((u, r) \in P.URA\)

For each \((r', p, y) \in P.PRPA\)

If \(x \neq y \land r' \preceq r\) then \(S = S \cup \{(u, p, x), (u, r), (r', p, y)\}\)

For each \((t, p, y) \in P.PTPA\)

For each \((r', t) \in P.PTA\)

If \(x \neq y \land r' \preceq r\) then

\(S = S \cup \{(u, p, x), (u, r), (r', t), (t, p, y)\}\)

//Find the administrative roles of conflict rules

For each rule \(s \in S\)

For \(i\) from 1 to \(n\)

If \(s \in P_i\) then \(S_c = S_c \cup \{(c, a_i)\}\)

(2) The conflicts can also be resolved in run time access check. In run time, there are several different strategies to resolve the conflicts. These strategies can be applied in different situations according to the practical requirements.

1) Pessimistic (least privilege): If one administrative role defines the rule denying the operation, the operation is denied.

2) Optimistic (largest privilege): If one administrative role defines the rule granting the operation, the operation is permitted.

3) Parent first: The rules created by parent administrative role are always used. If the administrative roles have no parent-child relationships, use other strategies.

4) Child first: The rules created by child administrative role are always used. If the administrative roles have no parent-child relationships, use other strategies.

5) Union: If several administrative roles define different administrative domains for their common child administrative role, then the administrative domain of this child administrative role is the union of these different administrative domains.

6) Mediator: The conflict is decided by other mediator, such as their common parent administrative role or other administrative roles they both trust.

III. IMPLEMENTATION AND EXPERIMENTS

A. Implementation in TiPLM

Product Lifecycle Management System (PLM) is a software platform which integrates peoples, processes and information in support of the collaborative creation, management, dissemination, and use of product definition information across the extended enterprise from concept to end of life. Figure 2 shows the access control mechanism in TiPLM.

The access control module includes Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Administration Point (PAP), Conflict Detector, Conflict Resolver and Attribute Finder sub-modules. The authorization policies are related with PDP and the administration policies are related with PAP. PEP constitutes the point where the policy decisions are actually enforced. PDP is responsible for decisions the access permission to the requested resources according to the authorization policies. Conflict Detector sub-module is to detect the potential conflicts after the access control policy has been changed. Conflict Resolver sub-module is responsible for giving a final decision when several policy rules are matched for a single request. Attribute Finder sub-module is to get the attribute of subjects and objects for computing the conditions and constraints in authorization rules. PAP is responsible for the interaction with administrative functions to manage the authorization and administration policies according to the administration policies.

The user interface of TiModeler is shown as Figure 3. The TiModeler used by administrative role
systems.

First, it is a great challenge to implement DA TRBAC as separation of duty (SOD), contexts, delegations etc. to support other important access control components, such as access check, authorization and administration aspects. We proposed an algorithm to detect access conflicts during design time and proposed several strategies to resolve these conflicts during run time. The average access check time for each operation is about 0.02 seconds.

Second, it is a great challenge to implement DA TRBAC as separation of duty (SOD), contexts, delegations etc. to support other important access control components, such as access check, authorization and administration aspects. We proposed an algorithm to detect access conflicts during design time and proposed several strategies to resolve these conflicts during run time. The average access check time for each operation is about 0.02 seconds.

IV. CONCLUSION AND FUTURE WORK

It is rather a great challenge to balance the competing goals of collaboration and security in collaborative environments. In this paper, we focus on the access control aspect to provide security for organizations in process collaboration environments. We proposed a domain based administration model for task-role based access control. This model combines access check, authorization and administration aspects of access control. We proposed an algorithm to detect the conflicts during the design time and proposed several different strategies to resolve these conflicts during run time access check. We also described in detail how to implement the model in the PLM product.

The work presented in this paper can be extended in several directions. First, the DATRBC model can be extended to support other important access control components, such as separation of duty (SOD), contexts, delegations etc. Second, it is a great challenge to implement DATRBC model in the integration systems which combine PLM with other enterprise information products (CAPP, ERP etc) to support the process collaborations among the heterogeneous systems.

ACKNOWLEDGMENT

The work is partially supported by the National Natural Science Foundation of China (Grant no. 60673008, Grant no. 60673122), the National High-Tech Research and Development Program of China (Grant No. 2007AA04Z135).

REFERENCES


